TFM S.M.A.R.T. Swim MApping, Reporting and Training

Máster en Big Data, Cloud and Analytics

October 17, 2020

MBIT School

Madrid Business Intelligence Technology

1 Máster en Big Data, Cloud & Analytics

1.1 TFM: S.MA.R.T. Swim MApping, Reporting & Training

1.1.1 Aprendizaje No Supervisado - Clustering

Fecha: Octubre 2020 Se define el estilo de visualización del notebook para que ocupe el 90% del ancho de la pantalla para facilitar el seguimiento.

```
<IPython.core.display.HTML object>
```

2 ANTECEDENTES

Nuestro partner tecnológico NBN23, especialista en aplicación de Big Data en entornos deportivos, ha desarrollado la aplicación nagi Smartpool, originalmente orientada a la seguridad en piscinas infantiles, con la intención de crear una plataforma para el seguimiento del entrenamiento en piscinas cubiertas que realizan tanto nadadores aficionados como profesionales. Ahora mismo esta aplicación se basa en una arquitectura *on-premises*, con las dificultades consiguientes en términos de escalabilidad e infraestructura, y solicitan una propuesta para recrear esta plataforma en un entorno *cloud* así como la obtención automática del estilo de nado de los usuarios, lo que supondría una gran ventaja competitiva a la hora de expandirse y ampliar su catálogo de aplicaciones.

3 OBJETIVOS DEL PROYECTO

El **INSIGHT** de este notebook es encontrar un método para **DETECTAR AUTOMÁTICAMENTE EL ESTILO DE NADO** de los usuarios de la aplicación, a partir de los datos **NO ETIQUETADOS** que se nos proporcionan. Se trata, por tanto, de un problema de **Aprendizaje No Supervisado (clustering)**, en el que se intentará clasificar las distintas sesiones de entrenamiento de los usuarios en función de los datos agregados más otros generados sintéticamente que nos ayuden a resolverlo.
Los estilos de nado que se van a intentar detectar son:

- `Front Crawl` : el también conocido como *Freestyle* es el más rápido, y en el que se recorre más espacio por cada brazada.
- `Breaststroke` : el estilo más lento, que se caracteriza también por tener la cabeza sumergida la mitad del tiempo.
- `Others` : en esta categoría se agrupan los otros dos estilos más importantes, *Butterfly* y *Backstroke*.

Para conseguir nuestro objetivo habrá que realizar una serie de transformaciones a los datos, así como una serie de hipótesis de trabajo basadas en estudios realizados a nadadores profesiones (ver anexos del TFM), así como en la observación empírica de los usuarios de los gimnasios.

4 IMPORTACIÓN DE LIBRERÍAS, DATOS Y FUNCIONES

4.1 Importación de Librerías

4.2 Definición de Funciones Auxiliares

Se definen dos funciones auxiliares que se van a utilizar en el notebook: - Una función lambda que extrae el identificador de un registo cuando comienza por *ObjectId*. - Una función para imputación de valores atípicos (outliers) mediante el método 6-sigma o de reducción de la variabilidad.

4.3 Importación de Datasets

Nuestro partner tecnológico nos ha entregado cinco archivos CSV correspondientes a los datos recogidos durante otros tantos días en dos de las piscinas donde tienen instalado su sistema *on-premises*. Estos datos recogen datos anonimizados de tiempo, posición e identificación de tag y sesión de nado:

- `_id` : identificador único de fila.
- `TagID` : identificador único del tag que lleva cada nadador y que manda los datos al sensor.
- `DrillID` : identificador único de la sesión de entrenamiento.
- `TimeStamp` : marca de tiempo (en milisegundos transcurridos desde 01-01-1970).
- `Position` : vector con las corrdenadas cartesianas de posición, tomando como origen el centro de la piscina.
- `Velocity` : vector que marca la velocidad en cada instante. Los sensores en la actualidad no recogen estos datos.
- `Acceleration` : vector que marca la aceleración en cada instante. Los sensores en la actualidad no recogen la acelerometría.

5 ANÁLISIS EXPLORATORIO Y AUDITORÍA DE DATOS

5.1 Primeras visualizaciones

En primer lugar, ordenamos el dataframe que hemos creado por `TagID` (usuario), `DrillID` (sesión de ejercicio) y `TimeStamp` (tiempo).

	_id	TimeStamp	DrillID	TagID	Position	Velocity	Acceleration
71763	5e2a96a8d38878036402296f	1579849383847	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71767	5e2a96a8d38878036402975	1579849383887	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71771	5e2a96a8d38878036402297a	1579849383927	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71775	5e2a96a8d38878036402297f	1579849383967	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71779	5e2a96a8d38878036402984	1579849384007	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71783	5e2a96a8d38878036402988	1579849384047	5e2a96a8d38878036402296a	a4da22e0a243	[8.95442339,-3.5120962000000002,0.2000000000000...	[0,0,0]	[0,0,0]
71787	5e2a96a9d38878036402298c	1579849384087	5e2a96a8d38878036402296a	a4da22e0a243	[8.916343615454545,-3.512871673566434,0.200000...	[0,0,0]	[0,0,0]
71791	5e2a96a9d38878036402992	1579849384127	5e2a96a8d38878036402296a	a4da22e0a243	[8.878263840909092,-3.5136471471328674,0.20000...	[0,0,0]	[0,0,0]
71795	5e2a96a9d38878036402997	1579849384167	5e2a96a8d38878036402296a	a4da22e0a243	[8.840184066363637,-3.514422620699301,0.200000...	[0,0,0]	[0,0,0]
71799	5e2a96a9d38878036402299d	1579849384207	5e2a96a8d38878036402296a	a4da22e0a243	[8.802104291818182,-3.5151980942657346,0.20000...	[0,0,0]	[0,0,0]

```
Número de registros: 18479898
Número de variables: 7
```

Es importante conocer sobre cuántos usuarios y sesiones de entrenamiento vamos a trabajar para obtener nuestro *insight*, así como para comprobar al final que todos los usuarios han sido etiquetados.

```
Número de TagID únicos -usuarios- en dataframe para clustering         : 31
Número de DrillID únicos -largos nadados- en dataframe para clustering : 4940
```

5.2 Registros Duplicados y Variables Únicas

Se comprueba si nuestro dataframe tiene registros duplicados. Esto es particularmente interesante cuando se reentrene el modelo con datos históricos agregados, ya que eliminará los elementos repetidos.

```
Número de registros duplicados del dataset ORIGINAL:  0
```

A continuación se comprueba si alguna de las variables tiene un único valor. En nuestro caso ya sabíamos de antemano que las columnas *Velocity* y *Acceleration* están siempre a cero, por lo que se eliminan del dataframe.

```
Las variables del dataset ORIGINAL con todos sus valores iguales son:  ['Velocity', 'Accele
```

5.3 Missing Values (Valores Ausentes)

Es importante chequear si nuestro dataframe tiene valores ausentes (*missing values*) que puedan provocar un error a la hora de generar nuevas variables sintéticas. Para eliminar dichos valores ausentes hay varias formas de imputación, la más sencilla mediante el método `dropna`, pero en nuestro vamos a optar por uno de los métodos que proporciona

la librería Pandas: `fillna` sustituye el valor NaN por 0. Este chequeo de la existencia de *missing values* habrá que volver a hacerla una vez creadas las variables sintéticas que vamos a necesitar para el *clustering*.

```
Número de valores ausentes del dataset ORIGINAL:  0
```

5.4 Creación de Variables Sintéticas

Se generan tres nuevas variables a partir de `Position`, que corresponden a las coordenadas X, Y y Z (hay que tener en cuenta que el eje de coordenadas se sitúa en el centro de la piscina). A continuación, se elimina del dataframe la variable original puesto que al ser de tipo *object* no sirve para hacer el clustering.
Ya se ha comentado que la aplicación nagi Smartpool nació para cubrir la necesidad que tenían algunas piscinas infantiles para prevenir ahogamientos, y de hecho se sigue utilizando mediante la computación del tiempo que pasa entre dos muestras consecutivas pertenecientes a una misma sesión de entrenamiento. El umbral es configurable, y una vez superado se manda una alerta sonora o visual al socorrista de la piscina para que compruebe si algo anormal está sucediendo. Por tanto, la coordenada Z que envían los tags a los sensores Bluetooth es inútil y se fija siempre a un valor constante, por lo que se procede a eliminar también del dataframe, una vez comprobado este hecho (tienes dos valores distintos, ya que son dos las piscinas de las que tenemos datos).

```
Número de valores únicos de la variable PositionZ: 2
```

Se crea una nueva variable `TimeStampDT`, en formato `yyyy-mm-dd hh:mm:ss`, a partir de `TimeStamp` para que la visualización temporal de las muestras sea más amigable (ambas variables se eliminarán del dataframe a la hora de prepararlo para el algoritmo de clustering).

```
Número de registros: 18479898
Número de variables: 7
```

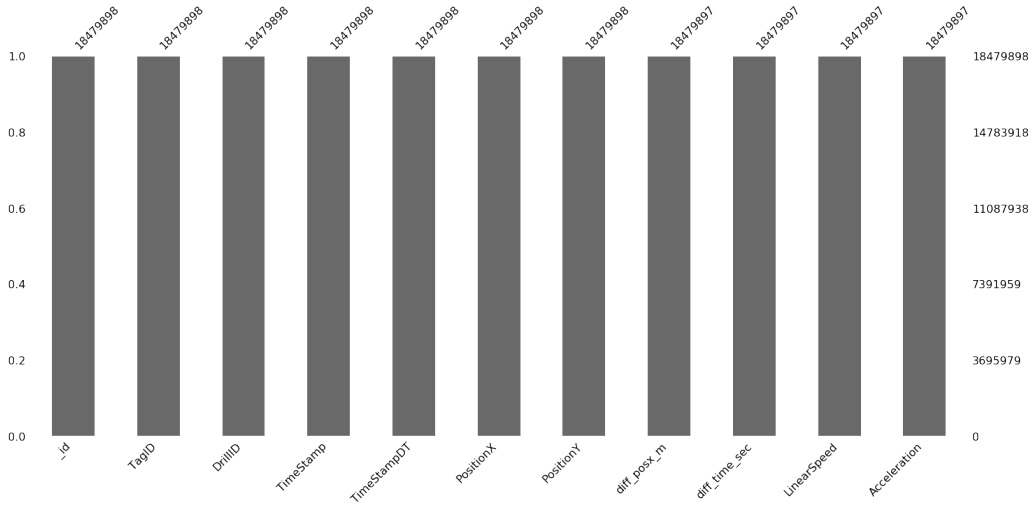
A continuación, se ordena el dataframe y se crea una primera copia de control.
Los datos que se van a utilizar para Machine Learning son de dos piscinas; una en USA que está abierta las 24h del día por lo que la distribución de las variables temporales `TimeStamp` y `TimeStampDT` en relación a la otra, situada en un gimnasio de Madrid con horario de 6am a 12pm, será distinta. Esta distribución temporal asimétrica a efectos de clustering no tiene ningún efecto.

Creamos otras dos variables sintéticas **FUNDAMENTALES** para el modelo de Aprendizaje No Supervisado que vamos a desarrollar, puesto que nuestra hipótesis de trabajo se va a basar en que el estilo de nado más rápido es el `Front Crawl/Freestyle` y el más lento el `Breaststroke`. Otros estilos de natación, como `Backstroke` o `Butterfly`, son mucho más complejos de predecir puesto que entran en juego más factores como el tiempo de inmersión y la acelerometría, datos que por el momento no son capturados por los sensores que llevan los nadadores (se propondrá en el apartado de conclusiones del TFM como mejora para refinar el modelo).
Por lo tanto, se añaden otras dos variables a nuestros dataframes de trabajo: `LinearSpeed`, velocidad lineal instantánea, y `Acceleration`, aceleración instantánea. Las fórmulas de la velocidad y aceleración son bien conocidas: incremento de la distancia entre el incremento del tiempo, e incremento de la distancia entre el cuadrado del incremento del tiempo, respectivamente.

```
Número de registros: 18479898
Número de variables: 11
```

5.5 Missing Values (NaNs) después de crear variables sintéticas

Utilizamos la librería `missingno` para visualizar en un gráfico de barras los *missing values* de nuestro dataframe. Se comprueba que los únicos que existen son los valores iniciales de las cuatro nuevas variables sintéticas, por lo que se pueden sustituir por 0 utilizando en método `fillna` que proporciona el paquete Pandas.



```
Número de valores ausentes del dataset ANTES de la imputación  :  4
Número de valores ausentes del dataset DESPUÉS de la imputación:  0
```

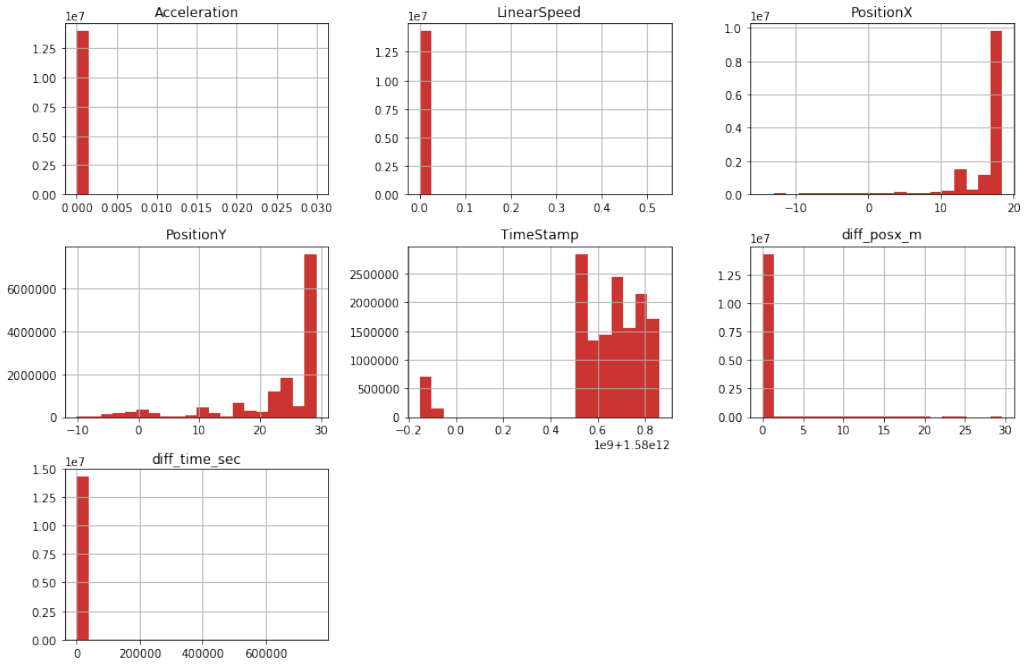
5.6 Análisis de Valores Atípicos (Outliers)

Consultando las plusmarcas mundiales de natación, hacemos un primer filtrado muy básico de las variables `LinearSpeed` y `Acceleration`, ya que se da por supuesto que ningún campeón del mundo va a entrenar a nuestros gimnasios.
De acuerdo a lo anterior, el récord del mundo en 100 metros braza femenino (el más lento de todos) está establecido en 1:04.13 minutos. Es decir, con esos datos la velocidad media es de aproximadadmente 1.56 m/s, y la aceleración sería inferior a 0.03 m/s2. Así que nuestra primera hipótesis de trabajo supone que ningún nadador entre los usuarios de nuestras piscinas va a superar esas marcas, por lo que se eliminan del dataframe. En caso de que nuestra aplicación tuviera éxito y se expandiera a otro tipo de piscinas más orientadas a entrenamiento de deportistas profesionales, este umbral habría que modificarlo.

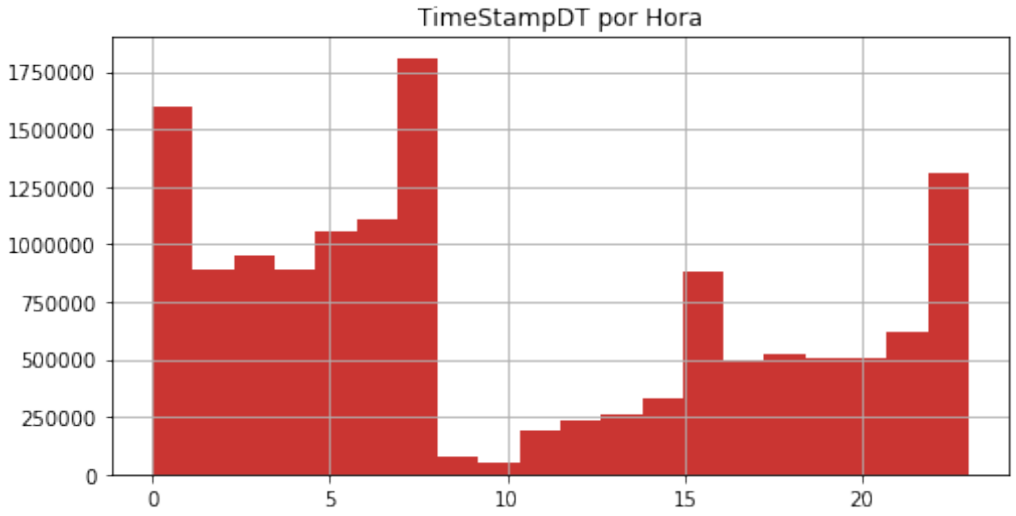
```
Número de registros: 14287443
Número de variables: 11
```

5.6.1 Representaciones Gráficas (Histogramas y Boxplots)

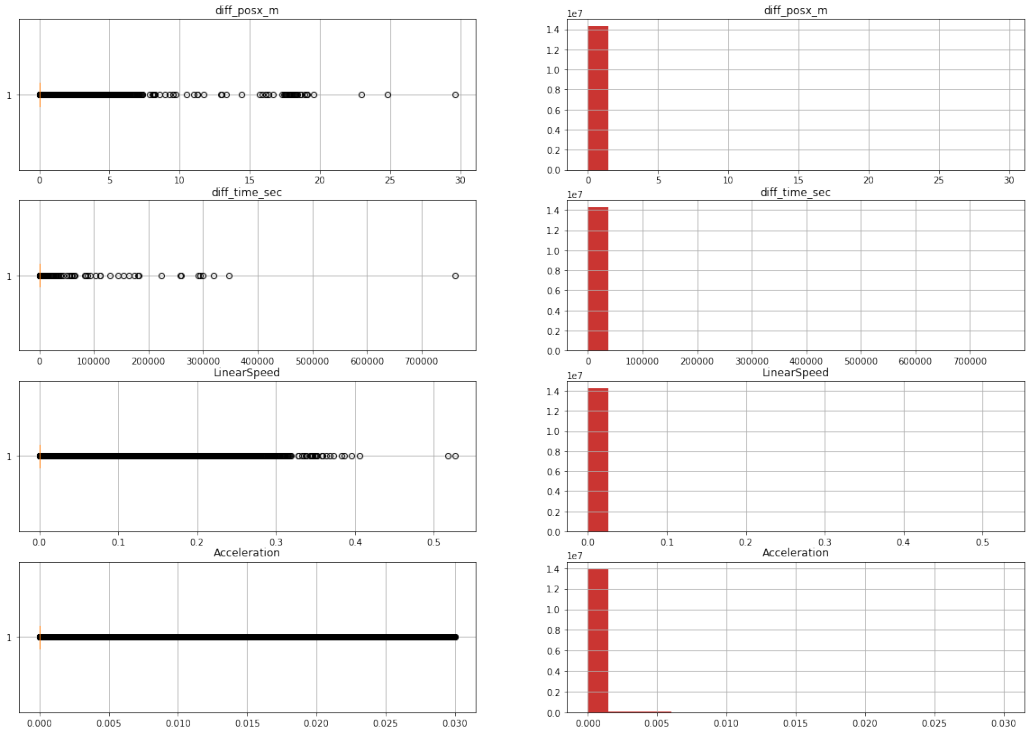
A continuación, con la ayuda de histogramas y boxplots, vamos a analizar la distribución de las variables de interés en cuanto a la imputación de valores atípicos (*outliers*) se trata. En primer lugar, observamos la distribución de las variables numéricas.



Comprobamos que la afluencia a la piscinas se prolonga las 24 horas (en USA este gimnasio está abierto todo el día).



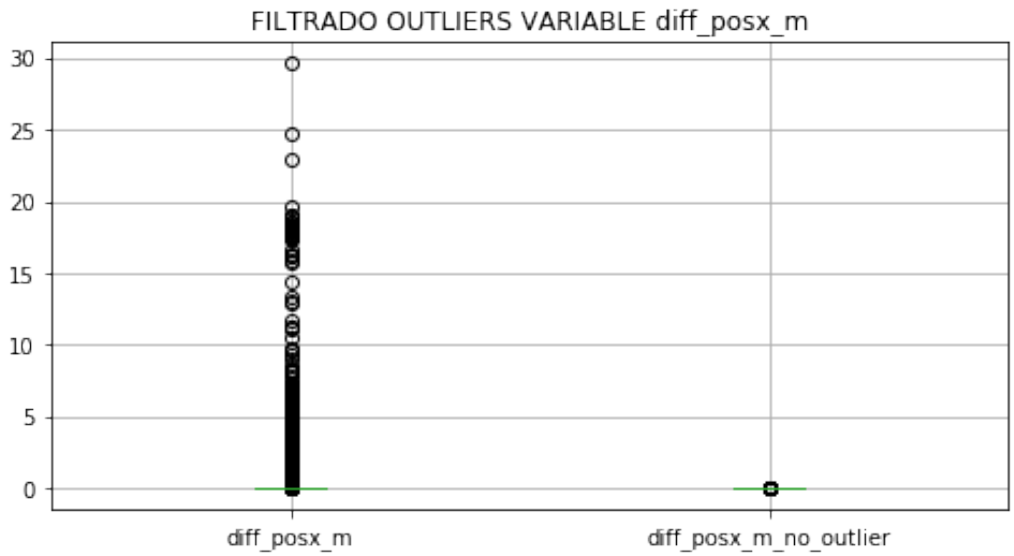
Una herramienta muy útil a la hora de ver los *outliers* son las representaciones combinadas de boxplot más histograma. En nuestro caso, como las variables que nos interesa filtrar son las sintéticas que hemos creado, definimos primero una lista con esas cuatro variables y luego ejecutamos los gráficos.

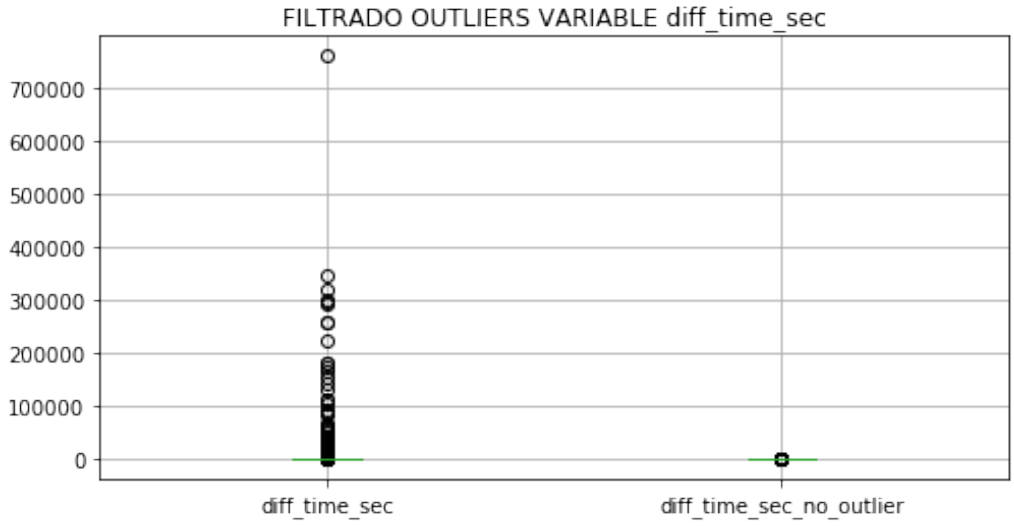


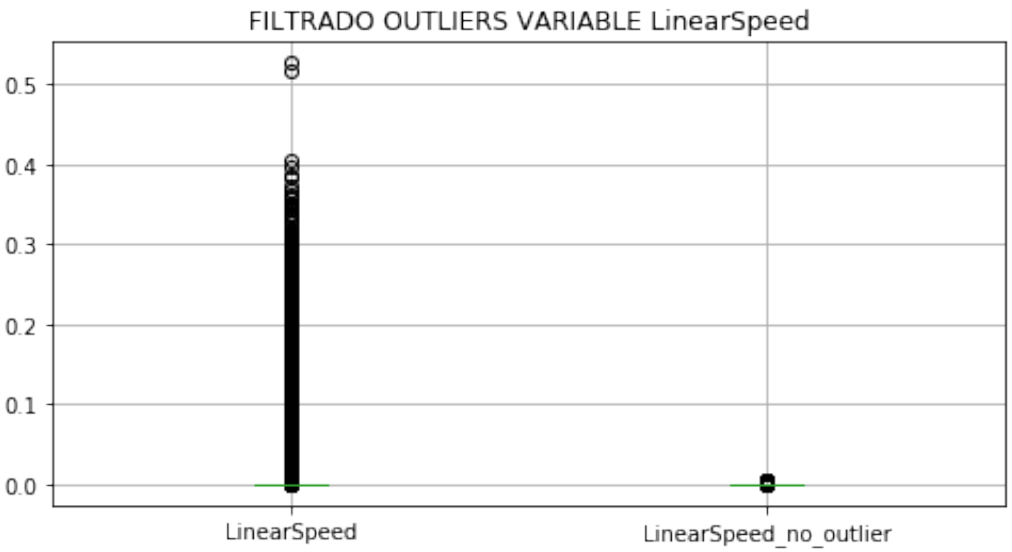
Observamos que estas cutaro variables presentan bastantes valores atípicos que habrá que filtrar; el origen de estos *outliers* se debe principalmente a que entre una sesión de entrenamiento `DrillID` y la siguiente de un mismo usuario `TagID` pueden pasar horas e incluso días.
Así que haciendo uso de la función definida en el primer apartado para la imputación de los valores atípicos:

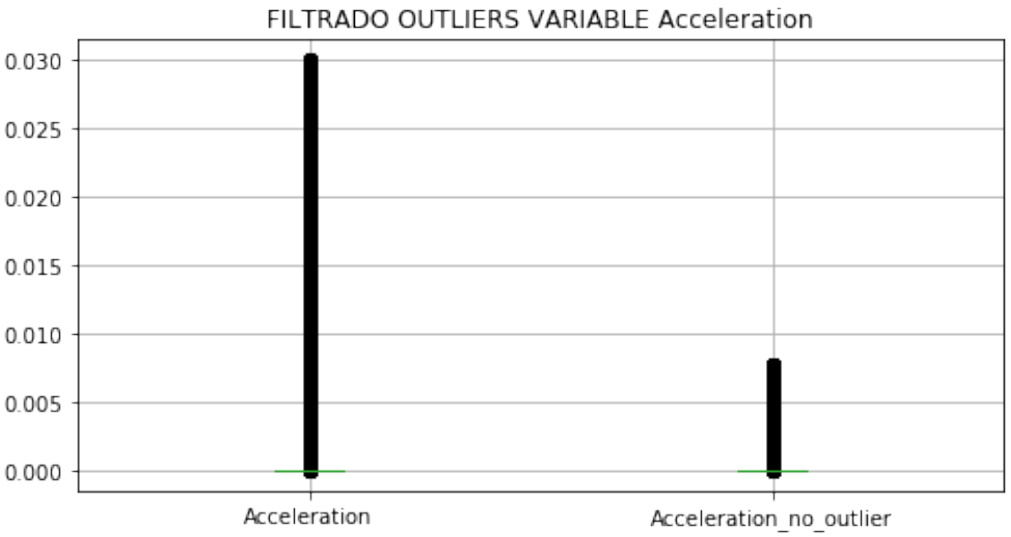
```
def remove_outliers_6sigma(X):
    return X[abs(X - np.mean(X)) < 3 * np.std(X)]
```

Comprobamos su efecto sobre las variables de interés `diff_posx_m`, `diff_time_sec`, `LinearSpeed` y `Acceleration`, puesto que los valores extremos de las otras variables corresponden a los límites horizontal y vertical de la piscina.


FILTRADO OUTLIERS VARIABLE diff_posx_m
30
25
20
15
10
5
0
diff_posx_m
diff_posx_m_no_outlier


FILTRADO OUTLIERS VARIABLE diff_time_sec
700000
600000
500000
400000
300000
200000
100000
0
diff_time_sec
diff_time_sec_no_outlier

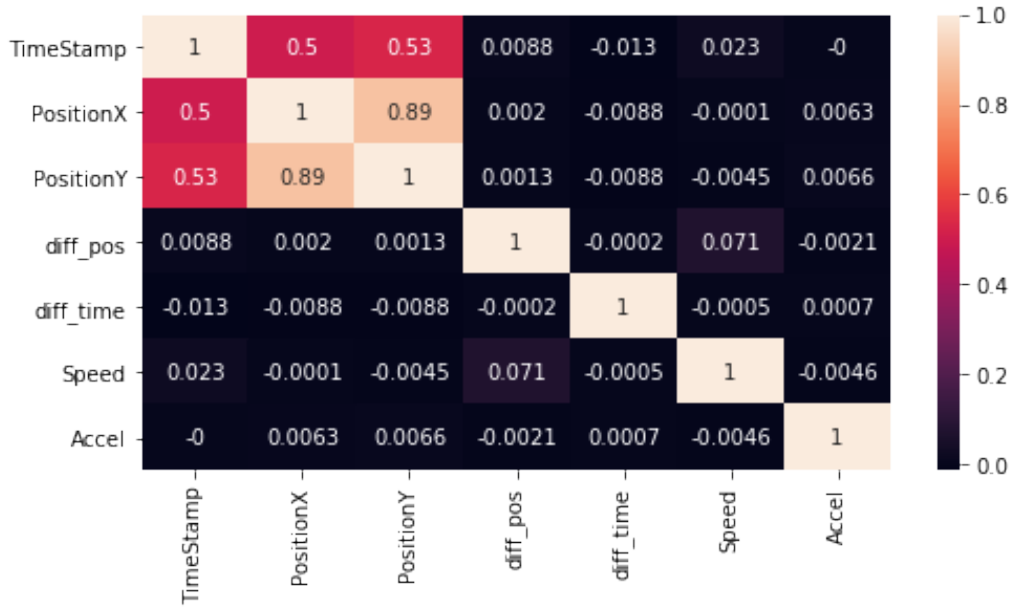




Una vez comprobada la efectividad de la función, pasamos a aplicarla a nuestro dataframe:

Para terminar este apartado se eliminan las variables filtradas, que ya no son útiles para el clustering, y se renombran las nuevas que se han generado al aplicar la función de imputación de *outliers*.

5.7 Matriz de Correlación



Comprobamos que la variable `PositionY` presenta un valor de correlación alto (superior al 80%) con respecto a la variable `PositionX`, por lo que podría ser considerada como un "falso predictor", así que se elimina del dataframe.

5.8 Agregación de Datos

Para poder hacer un Aprendizaje No Supervisado efectivo, necesitamos tener los datos agrupados por usuario `TagID` y sesión de ejercicio `DrillID`. A su vez, se realiza una agregación de estos datos agrupados para obtener distintas métricas de interés (principalmente máximo, mínimo y media aritmética).
Vamos a realizar de nuevo una breve descripción de las variables que tenemos en este momento en el dataframe:

- `TagID` : identificador único de usuario.

- `DrillID` : identificador único de sesión de entrenamiento.

- `TimeStamp` : marca de tiempo en milisegundos.
- `TimeStampDT` : marca de tiempo en formato `yyyy-mm-dd hh:mm:ss`.
- `PositionX` : coordenada cartesiana X tomando como referencia el centro de la piscina.
- `diff_pos` : incremento de distancia lineal entre dos muestras consecutivas.
- `diff_time` : incremento temporal entre dos muestras consecutivas.
- `Speed` : velocidad lineal instantánea.
- `Accel` : aceleración instantánea.

Además de estas variables ya conocidas se va a generar otra de tipo booleano, que nos va a indicar si el registro se considera valor atípico en función de los valores máximos y mínimos de la coordenada X.

- `Outlier_PositionX` : indicador de valor atípico en función de los datos de PositionX.

En este punto es interesante comprobar los datos de los que vamos a disponer para hacer clustering en comparación con los que tiene el dataframe original sin hacerle ninguna transformación.

```
Número de registros del dataframe original     : 18479898
Número de registros del dataframe agregado     : 4457
Número de variables del dataframe original     : 7
Número de variables del dataframe agregado     : 21

Número de TagID únicos del dataframe original  : 31
Número de TagID únicos del dataframe agregado  : 31
Número de DrillID únicos del dataframe original: 4940
Número de DrillID únicos del dataframe agregado: 4455
```

Nota: es importante resaltar que el número de usuarios entre el dataframe original y el agregado una vez hechas todas las transformaciones e imputaciones es el mismo, por lo que a la hora de presentar resultados del modelo TODOS los usuarios iniciales tendrán al menos una predicción de su estilo de nado.

```
DrillID -sesiones- eliminados después de imputar outliers: 2909359
```

Nota: se pierden casi tres millones de registros (de los más de 18 que teníamos al principio), una vez imputados valores ausentes y atípicos.

```
Número de DrillID únicos del dataframe original      : 4940
Número de DrillID únicos después de imputar NaNs     : 4940
Número de DrillID únicos después de eliminar outliers: 4455
Número de DrillID únicos del dataframe agregado      : 4455
```

Nota: se pierden alrededor de 500 sesiones de ejercicio al preparar el dataframe para Machine Learning. Esto se traduce en que habrá sesiones que haya que etiquetar como estilo de nado `unknown`.

5.9 Eliminación de variables (identificadores únicos, marcas de tiempo, valores atípicos)

Eliminamos los identificadores únicos y las variables temporales, ya que son inútiles para nuestro problema de clasificación no supervisado.

```
C:\Users\calfo\Anaconda3\envs\env_ml\lib\site-packages\pandas\core\generic.py:3936: Perform
  obj = obj._drop_axis(labels, axis, level=level, errors=errors)
```

Asimismo, se considera un valor atípico aquel cuyo producto entre el máximo y el mínimo de la variable `PositionX` sea mayor igual a cero, ya que no habrá pasado por el origen de coordenadas, a todos los efectos nuestro centro de la piscina.
Se le pasa a nuestro dataframe una máscara que elimine los registros considerados como *outliers* referidos a la variable `PositionX`.

5.10 Flatenning de variables del dataframe agregado

6 APRENDIZAJE NO-SUPERVISADO (CLUSTERING)

El `clustering` (o *agrupamiento* en castellano) es un tipo de aprendizaje automático no-supervisado que consiste en la agrupación automática de datos. La librería de Python `scikit-learn` ofrece implementaciones eficientes de varias técnicas de agrupamiento y ofrece una tabla que puede ayudar a comparar diferentes algoritmos de *clustering*. Dicha comparación se realiza en función de los parámetros necesarios, su escalabilidad, caso de uso y geometría (métrica usada).
El algoritmo de *clustering* más usado es `K-Means`, ya que posee una muy buena escalabilidad con la cantidad de datos. Para utilizar `K-Means` se debe especificar el número K de grupos o *clusters* que queremos encontrar. El algoritmo sigue entonces los siguientes pasos:

- `Inicialización` : se elige la localización de los centroides de los K grupos aleatoriamente.

- `Asignación` : se asigna cada dato al centroide más cercano.
- `Actualización` : se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo.
- Los pasos 2 y 3 se siguen iterativamente hasta que no haya más cambios.

```
Número de registros para clustering: 4445
Número de variables para clustering: 15
```

6.1 Hipótesis de Trabajo para Aprendizaje No Supervisado S.Ma.R.T.

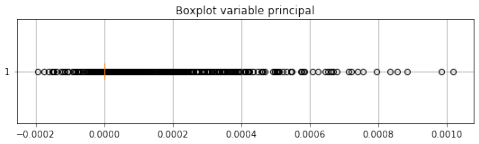
Se genera una nueva variable sintética `classifier_`, de tipo logarítmico y directamente proporcional a la velocidad media e inversamente proporcional al producto de la aceleración por el incremento de distancia. Dicha variable me va a permitir discriminar los estilos de nado, puesto que el más rápido es el `Front Crawl/Freestyle`, mientras que el más lento (con menor aceleración y distancia recorrida) es el `BreastStroke`. Para otros estilos (*Backstroke* o espalda, y *Butterfly* o mariposa) entran en juego más factores como el tiempo en el que el nadador permanece sumergido, por lo que queda como propuesta de mejora que los sensores recojan más datos biométricos para refinar el *clustering*.
El cálculo de esta variable se ha realizado teniendo en cuenta las conclusiones de dos estudios científicos realizados sobre nadadores profesionales y que se pueden encontrar adjuntos a la memoria de este TFM.

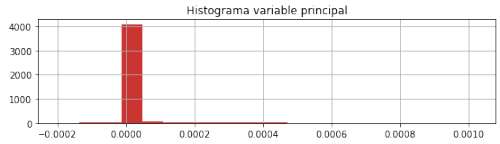
1. Accelerometer Profile Recognition of Swimming Strokes
2. Inertial Sensor Technology for Elite SwimmingPerformance Analysis

6.2 Boxplot + Histograma Variable Clasificadora

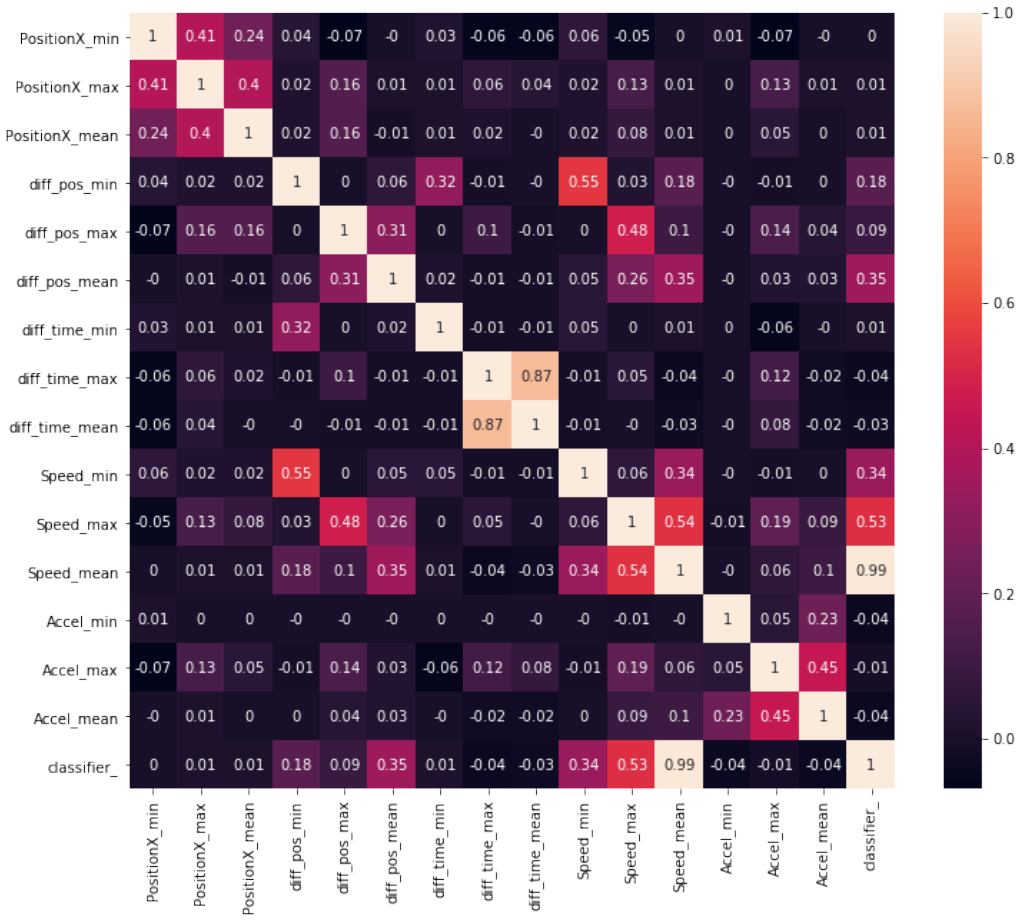
Se revisa la distribución de la nueva variable de clasificación, aunque no se eliminan los posibles *outliers* ya que seguramente sean los que nos permitan discriminar mejor los estilos de natación.

```
Text(0.5, 1.0, 'Histograma variable principal')
```





6.3 Matriz de Correlación



Nota: se comprueba que las variables `diff_time_max` y `Speed_mean` presentan altos valores de correlación (> 80%), por lo que no valen como predictores y se eliminan del dataframe.

6.4 Algoritmo de Clustering K-Means

Una de las recomendaciones más interesantes antes de aplicar un algoritmo `K-Means` es la **normalización** de los datos: en este contexto, «normalizar» se refiere a que los valores de cada atributo estén en escalas similares. Normalizar ayuda al *clustering* ya que los grupos se forman a partir de distancias y, si hay atributos con escalas muy diferentes, los

de escala mayor dominarán las distancias. Una de las técnicas más comunes de normalización (y que se va a utilizar para nuestro modelo) es re-escalar cada atributo en el rango [0, 1].
Según se ha indicado anteriormente, nuestro modelo `K-Means` va a tener tres clusters, correspondientes a los estilos predominantes de natación en piscina cubierta:

- `Front Crawl` : también conocido como *Freestyle*, es el más rápido, con el que se recorre más distancia por cada brazada, y el predominante en las piscinas de entrenamiento según se ha observado.
- `Breaststroke` : el estilo más lento, caracterizado por tener la cabeza sumergida la mitad del tiempo, y el minoritario en las piscinas cubiertas a excepción de ciertos sectores de edad (i.e. el estilo preferido por gente mayor).
- `Others` : en esta categoría se agruparán el resto de estilos más complejos de detectar, *Butterfly* y *Backstroke*.

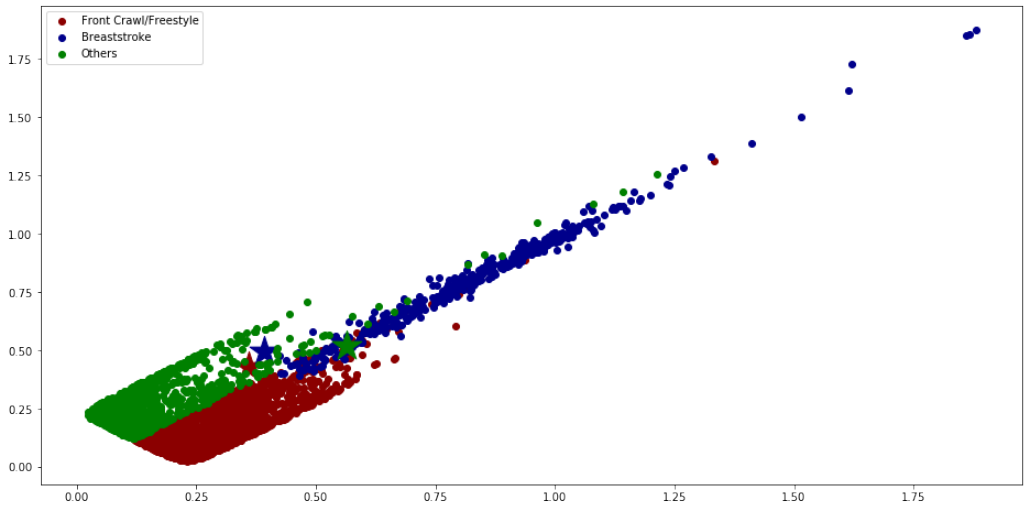
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

Nuestra segunda hipótesis de trabajo es que el cluster cuyo centroide presente un valor máximo de la variable clasificadora que hemos definido previamente corresponderá a sesiones de nado en que el estilo predominante ha sido `Front Crawl` o `Freestyle`, mientras que el centroide que presente el mínimo corresponderá a un estilo `Breaststroke`. La agrupación restante la etiquetaremos como `Others`.

```
The min value of classifier function corresponds to cluster 1
The max value of classifier function corresponds to cluster 0
```

```
['Front Crawl/Freestyle', 'Breaststroke', 'Others']
```

Una vez que tenemos a qué estilo de nado corresponde cada clúster, es el momento de representarlo gráficamente. Para ello, lo primero que hay que realizar es una transformación que ajuste el modelo a nuestro dataframe para poder hacer una visualización 2D de los resultados.



OBSERVACIONES: es interesante comprobar cómo hay cierto solape entre los clústers que corresponden a `Freestyle` y `Others`. Esto se puede deber a que el estilo `Breaststroke` es, con mucha diferencia, el más lento de todos, por lo cual es más fácil discriminarlo.
Para intentar ver mejor los tres clústers podemos hacer una proyección 3D mediante la función `go.Scatter3d` de la librería `plotly`.
NOTA: la visualización 3D confirma lo observado anteriormente.

7 ÚLTIMOS PASOS Y CONCLUSIONES

El último paso de toda la parte de Data Science y Machine Learning consiste en la preparación del archivo CSV con los resultados: va a consistir en un dataset con únicamente las tres variables que nos interesan, `TagID` (usuario), `DrillID` (sesión de entrenamiento) y `Predicted_Style` (estilo detectado de nado).

```
Predicted_Style
Breaststroke              402
Front Crawl/Freestyle    2884
Others                   1159
Name: classifier_, dtype: int64
```

OBSERVACIONES: estos resultados son muy interesantes, porque correlan bastante bien con la realidad observada a pie de piscina, donde entre un 60-70% de los usuarios nadan

a `Front Crawl` (estilo libre), mientras que un 10-15% lo hacen a `Breaststroke` (braza). El resto suele practicar estilos menos ortodoxos como `Butterfly` (mariposa), `Backstroke` (espalda), o combinaciones de varios.

7.1 S.Ma.R.T. Swimming Style Insight: Resultados

Volcamos el dataframe con los resultados proporcionados por el modelo de Machine Learning en un archivo CSV, que pueda ser explotado desde otras aplicaciones de visualización de datos, tanto en local como en la nube.

7.2 Exportación del Modelo de Clustering S.Ma.R.T.

Para poder utilizar nuestro modelo entrenado en el entorno `cloud` que se ha diseñado en AWS, es necesario exportarlo para que pueda ser llamado desde, por ejemplo, un script Python o una libreta de `Databricks`. Para ello se utiliza el método `dump` de la librería `joblib`.

```
['smart_model.pkl']
```

7.3 CONCLUSIONES

- El objetivo principal `insight` del proyecto se ha conseguido: **detectar automáticamente el estilo de nado** de una determinada sesión de entrenamiento con los datos proporcionados por nuestro partner.
- Al no tener los datos etiquetados previamente, la única forma de **validar** el modelo sería mediante un etiquetado manual que corroborara los resultados de este proyecto.
- Las transformaciones realizadas al conjunto original de datos provocan que se pierda algo menos del 10% de las sesiones de entrenamiento para realizar la predicción, por lo que habrá algunos datos que se quedarán sin clasificar correctamente.
- Al no tener los sensores activadas las capacidades de detección de velocidad y acelerometría se pierde mucha información valiosa para nuestro modelo. Si se dispusiera de esos datos, el modelo se podría depurar y no depender de la creación de variables sintéticas, lo que ayudaría en última instancia a no perder tantos registros.
- Para integrar este modelo en nuestra arqueitectura `cloud`, únicamente habrá que cambiar el origen de los datos desde un directorio local a un archivo de datos almacenado en AWS S3. Para ello, se van a generar tres scripts de Python:

`data_preparation` que cubrirá la parte de preparación del dataset para su explotación, `clustering_insight` que generará un resultado en base al modelo `K-Means` exportado y unos datos de entrada, y `model_training` que se encargará de reentrenar el modelo de *clustering* con los nuevos datos que vayan llegando.

```
Execution took: 0:10:36 secs (wall clock time)
```